

REMARKS/ARGUMENTS

Claims 6-11 are now in the application. Claims 1-5 are canceled without prejudice and some of that subject matter is presented in clearer form in new claims 6-11. Applicants respectfully request reconsideration of the application, as amended, in view of the following remarks.

Applicants have amended the drawings and the specification to address the Examiner's objections and rejection. Applicants have amended paragraph [0112] to include the following new reference numbers for Figure 12: reference numbers 150, 152, and 154 refer to the rectangles; reference numbers 156, 158, 160, and 162 refer to the ovals; and reference numbers 164, 166, 168, 170, 172, 174, 176 and 176 refer to the arrows. The Replacement Sheet for Figure 12 showing those reference signs is attached hereto. The Replacement Sheet for Figure 1 shows reference signs 10, 12, 14, 16, 18, and 20, and the Replacement Sheet for Figure 13 shows the reference signs 134, 136, 138, 140, 142 and 144. Applicants have amended paragraph [0001] to include the missing serial numbers and to amend the specification to incorporate by reference the contents of provisional patent application Serial No. 60/464,019, to which the present application claims priority under 35 U.S.C. §119(e).

Claims 1-5 were rejected under 35 U.S.C. §101. These claims have been canceled and the new claims are recited using relevant language that is similar to that used in related U.S. patent application "Method and System for Detecting Vulnerabilities in Source Code," Serial No. 10/825,007, which faced a similar rejection. That case has since been determined to be recited in an appropriate manner. Consequently, Applicants believe the new claims are likewise directed to proper subject matter under 35 U.S.C. §101.

Claims 1-5 were provisionally rejected under obviousness-type double patenting of co-pending application 10/825,007. Even though claims 1-5 are canceled, a terminal disclaimer will be filed in conjunction with this paper to obviate any provisional rejection of the new claims.

Claims 1-5 were rejected under 35 U.S.C. §102 as being anticipated by Viega et al.

Viega, while discussing race condition analysis, does not disclose or suggest the approach recited in the amended claims. In particular, Viega does not consider control flow in detecting race conditions. In addition, Viega does not perform a semantic analysis of the code to determine the control flow or to resolve routine arguments precisely.

Section 4.1 of Viega discusses how it breaks the source code text up into a stream of tokens. The stream of tokens reflects the order of appearance for the words in the source code listing. In section 4.4.2, Viega provides a description of its race condition analysis. Variable names are mapped to their uses in function calls. If this list contains two calls related to the TOCTOU (Time-of-Check/Time-of-Use) race condition, Viega reports it as a vulnerability.

The Viega list of tokens is not reflective of run-time control flow. (It is well known that a program may execute in a different order than the linear sequential expression of the source file as a result of breaks in control flow from branching, jumping, routine calls, etc.) Likewise, the Viega list of tokens is not reflective of semantic meaning of the program.

Paragraph 3 in section 4.1 of Viega states that no pre-processing is done. (This is also made clear in section 3.1.2 when Viega states they wanted to work in an editing/programming environment like EMACs, i.e., trying to flag errors during creation of the program, before the program was even complete and thus factually not amenable to semantic processing.) Section 4.4.2 is explicit that Viega “fails to take control flow into account.”

Thus, in Viega, the words are not (and cannot be) mapped to any semantic understanding of the program (including control flow and data flow, or of the routine arguments for example if expressions or aliases are used). This is so because in order to glean semantic meaning the words would have to be expanded to determine the real C language tokens.

For example, if a program contained the word “check” and that word should be expanded into the following C language statement

```
if (stat(filename)) return;
```

(for example via a define statement) Viega would have no knowledge of this expansion or of the control flow implied in the “if” statement. It would only see the word “check.” Viega would not understand that “check” should be analyzed as part of a TOCTOU vulnerability.

Since Viega does not consider control flow, if the following code were encountered, Viega would identify it is a vulnerability even though it isn’t one, because the stat() call must precede the fopen() call for a vulnerability to occur.

```
func( char * filename)  
{  
    fopen( filename,... )  
    stat( filename,... )  
}
```

Likewise, if Viega encountered code like the following, it would identify the situation as a race condition because the same variable name is used (i.e., “filename”), even though in fact different files are at issue. This is so because Viega does not resolve the semantics of the routine arguments to determine that different files in fact are involved.

```
func()  
  
{  
  
    char * filename = "file1";  
  
    stat (filename)  
  
    {  
  
        char * filename = "file2"  
  
        fopen (filename,...)  
  
    }  
  
}
```

Furthermore, if routines referred to the same computer file but one routine used a variable name and the other routine used an expression-reference that resolved to the same file, Viega would miss this situation because it does not resolve the semantics of the expressions and instead relies on a lexical analysis of variable name matching.

In contrast, the amended claims clearly recite that the control flow is considered. The control flow is modeled and considered in determining if a race condition exists. Claim 6 for example recites that the source code listing is analyzed to create computer models of the control flow to indicate the run-time sequence in which routine calls will be invoked and to create computer models of said arguments for the routine calls. It also recites that the control flow computer models are used to determine a run-time sequence of execution of a pair of routine calls, and to determine if the second routine to be executed has an argument referring to a file that is also referred to by an argument of the first routine to be executed. (See paragraphs [106] – [114] for support for the claims.)

None of this is taught or suggested by Viega.

In addition, the amended claims (e.g., claim 7) recite that the data flow is modeled to resolve the expression- references and operand-references to computer files that are arguments to the routines. In this way, more precise detection may occur as aliases will be understood, and likewise variables with the same name but which in fact are different (e.g., because of different scope) will be understood.

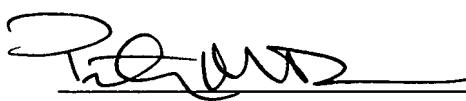
Applicants attempted to capture the features described above in the original claims (specifically the use of control flow in detecting race conditions), but it seems that those claims may have received an unintended interpretation. Applicants believe that the present amendments will avoid the unintended interpretations and should make these distinctive aspects more apparent.

For the reasons stated above, we believe that the claims are allowable.

The Commissioner is hereby authorized to charge any required fees to our Deposit Account No. 08-0219. Please apply any charges not covered, or any credits, to Deposit Account No. 08-0219.

Respectfully submitted,

Date: June 12, 2007



Peter M. Dichiara
Reg. No. 38,005

Wilmer Cutler Pickering Hale and Dorr LLP
60 State Street
Boston, MA 02109
Telephone: (617) 526-6466
Facsimile: (617) 526-5000

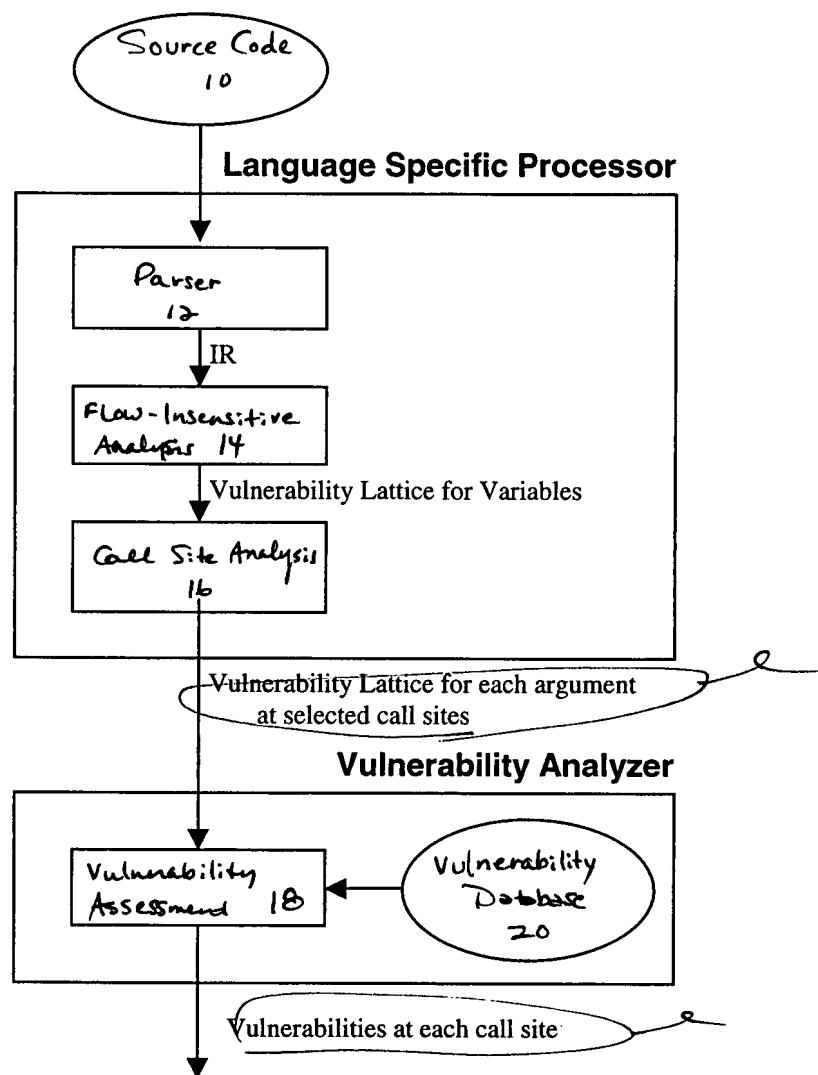


FIGURE 1

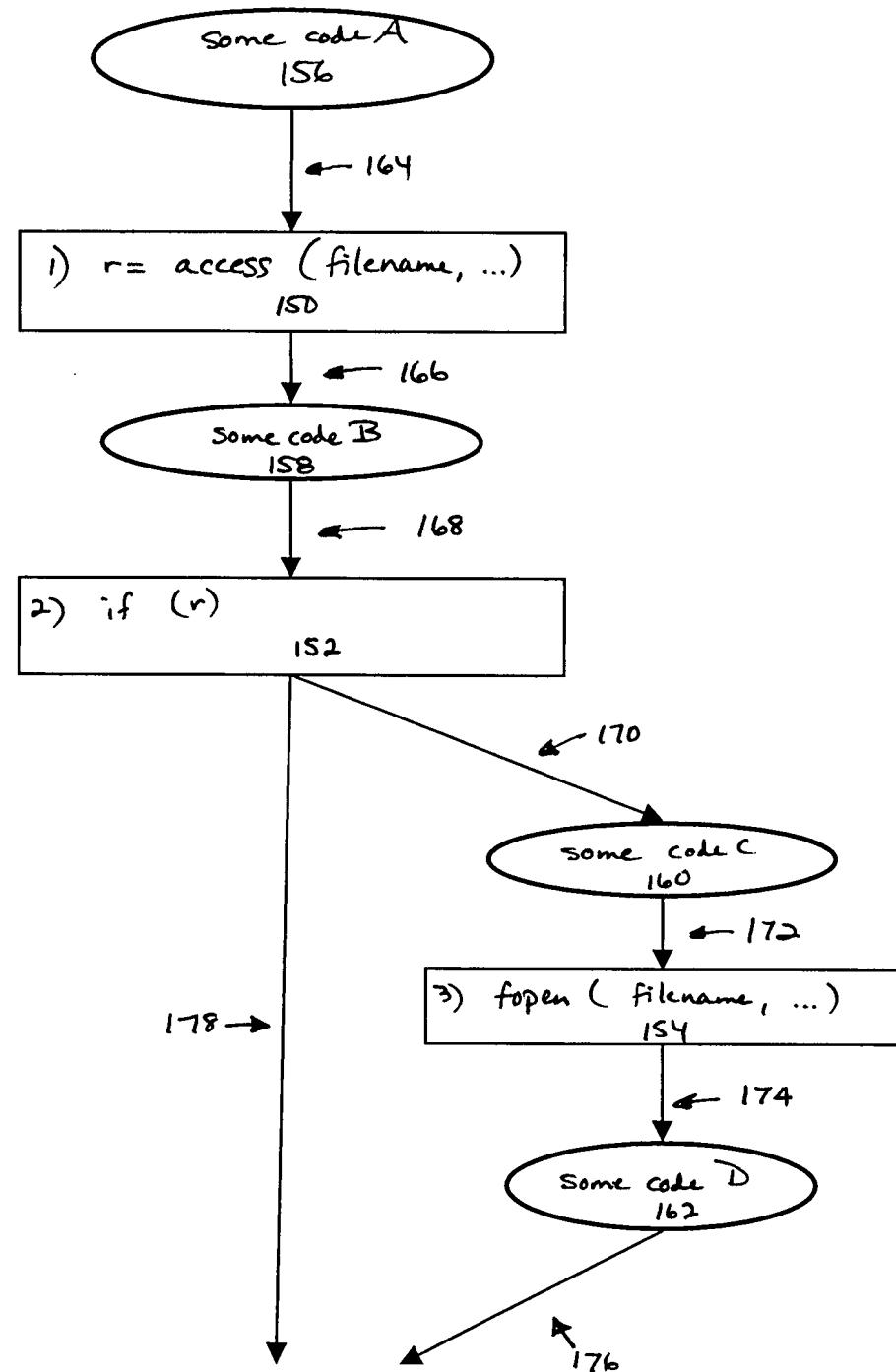


FIGURE 12

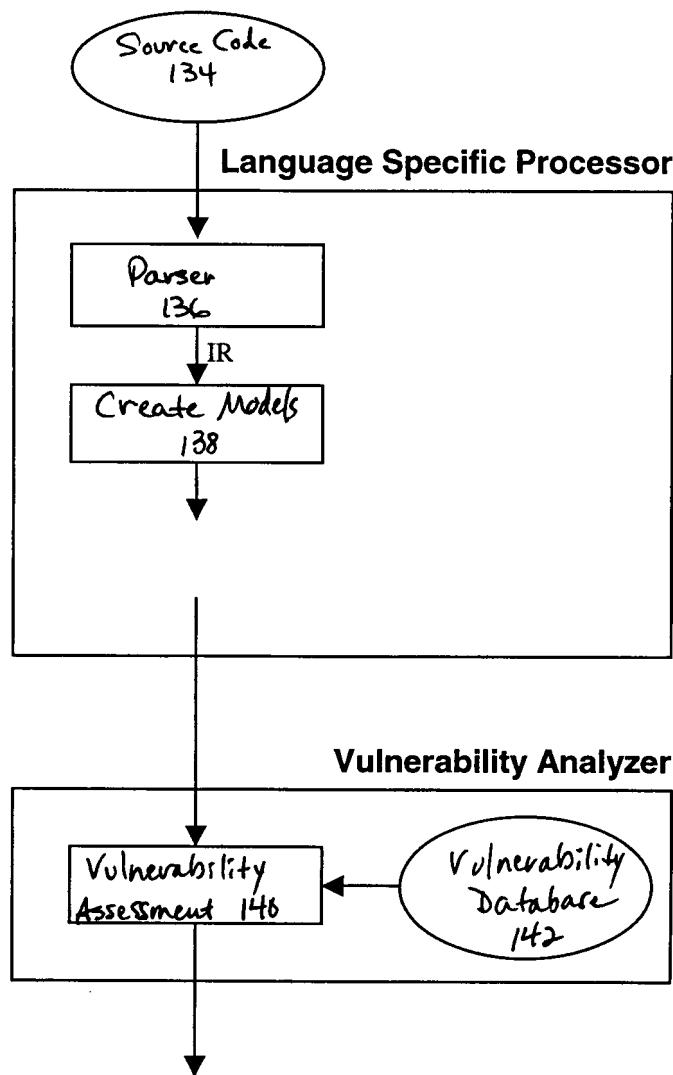


FIGURE 13